Rangkuman Pemrograman Berbasi Objek Pertemuan 1-6

Nama: Mohammad Ramadhan Zainoor

Kelas: 2IA11

NPM :50422914

Pertemuan 1-2:

Pemrograman berorientasi objek (OOP) adalah paradigma pemrograman yang menekankan penggunaan objek untuk memecah masalah program. Objek adalah entitas yang memiliki atribut, perilaku, dan kadang-kadang keadaan. OOP mulai ditemukan pada tahun 1960 sebagai perkembangan dari program terstruktur yang dikembangkan dari bahasa C dan Pascal. Program terstruktur memungkinkan penulisan program yang kompleks menjadi lebih mudah.

Pemrograman berorientasi objek (OOP) fokus pada objek-objek yang dapat menyelesaikan suatu masalah, bukan pada cara spesifik untuk menyelesaikannya. OOP bekerja optimal ketika dikombinasikan dengan proses Analisis dan Perancangan Berorientasi Objek (OOAD).

Pemrograman Berbasis Objek menggabungkan fungsi dan data dalam kelas atau objek, dengan ciri-ciri Encapsulation, Inheritance, dan Polymorphism. Struktur programnya ringkas dengan membuat objek dan kelas, serta sangat reusable karena objek dan kelas dapat digunakan berulang kali, menghemat ruang memori.

Pemrograman Berbasis Objek efektif digunakan untuk menyelesaikan masalah besar karena terdiri dari kelas-kelas yang memisahkan kode program menjadi kelompok-kelompok kecil sesuai fungsinya. Meskipun sulit di awal karena harus membuat kelas, namun setelahnya akan terasa mudah dan cepat. Eksekusinya lebih cepat karena dieksekusi bersamaan, dengan program hanya mengatur objek, propertinya, dan metodenya.

Class merupakan cetak biru atau blueprint dari objek. Digunakan untuk membuat kerangka dasar objek. Dalam analogi kendaraan, class bisa dibandingkan dengan kendaraan itu sendiri. Kendaraan memiliki ciri-ciri seperti merk, mesin, ban, dan lain-lain yang menandakan bahwa itu adalah kendaraan. Selain itu, kendaraan juga bisa melakukan tindakan seperti menghidupkan atau mematikan mesin.

```
// Deklarasi Class
public class Kendaraan{
    // Ini Adalah Class, semua konstruktor, variabel, method berada disini
}
```

Objek adalah entitas unik yang dapat merepresentasikan sesuatu di dunia nyata, seperti manusia, mobil, atau rekening bank. Mirip dengan rekaman dalam sistem

berkas, objek memiliki atribut dan metode yang mendefinisikan karakteristik dan perilakunya. Objek didefinisikan berdasarkan namanya atau menggunakan kata benda, dan bisa merepresentasikan kenyataan, konsep, atau visualisasi seperti huruf dalam sebuah font.

Dua karakteristik utama dari sebuah objek adalah:

1. Atribut (State): Ini adalah status dari objek yang mendefinisikan kondisinya pada suatu waktu tertentu. Contohnya, sebuah sepeda memiliki atribut seperti pedal, roda, jeruji, dan warna.
2. Tingkah Laku (Behaviour): Ini adalah aksi atau perilaku yang dapat dilakukan oleh objek. Contohnya, tingkah laku sebuah sepeda mencakup kecepatan naik, kecepatan turun, dan perpindahan gigi.

Dalam pengembangan perangkat lunak berorientasi objek, atribut dari objek disimpan dalam variabel, sedangkan informasi mengenai tingkah laku disimpan dalam metode atau fungsi-prosedur.

Untuk membuat objek, bisa menggunakan kata kunci new yang digunakan untuk membuat objek baru, selanjutnya menentukan 3 langkah untuk membuat sebuah objek. Yaitu: mendeklarasikan variable, membuat objek baru (Instansiasi) dan pemanggilan konstruktor.

```java
public class Kendaraan{

    // Konstruktor Dengan Parameter
    public Kendaraan(String nama){
        System.out.println("Nama Kendaraannya Adalah "+ nama);
    }

    public static void main(String[]args){

        // Perintah untuk membuat objek jenis
        Kendaraan jenis = new Kendaraan("Pesawat Terbang");

    }

}

// Output = Nama Kendaraannya Adalah Pesawat Terbang
```

Pertemuan 3-6:

Inheritance adalah konsep dalam pemrograman berorientasi objek di mana class baru dapat dibuat dari class yang sudah ada, dengan menambahkan atau memodifikasi fungsionalitasnya. Terdapat dua jenis inheritance: tunggal (single) dan ganda (multiple), namun Java hanya mendukung pewarisan tunggal karena kompleksitasnya. Dalam inheritance, constructor tidak diwariskan secara otomatis ke subclass, kecuali jika menggunakan perintah super.

Encapsulation adalah cara untuk melindungi property (atribut) atau method tertentu dari sebuah kelas agar tidak sembarangan diakses dan dimodifikasi oleh bagian program lainnya. Accesors adalah cara untuk melindungi data dengan menggunakan access modifiers (hak akses), yang terdiri dari:

No	Modifier	Pada class dan interface	Pada method dan variabel
1	Default (tidak ada modifier)	Dapat diakses oleh yang sepaket	Diwarisi oleh subkelas dipaket yang sama, dapat diakses oleh method-method yang sepaket
2	Public	Dapat diakses dimanapun	Diwarisi oleh subkelasnya, dapat diakses dimanapun
3	Protected	Tidak bisa diterapkan	Diwarisi oleh subkelasnya, dapat diakses oleh method-method yang sepaket
4	private	Tidak bisa diterapkan	Tidak dapat dikases dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri

Aksesabilitas	public	private	protected	default
Dari kelas yang sama	Ya	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak	Tidak
Dari subkelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari subkelas di luar paket	Ya	Tidak	Ya	Tidak

Polymorphism

- **Overloading** adalah diperbolehkannya dalam sebuah class memiliki lebih dari satu nama function/method yang sama tetapi memiliki parameter/argument yang berbeda.

```
1  public class Overloading {
2      public void Tampil(){
3          System.out.println("I love Java");
4      }
5      public void Tampil(int i){
6          System.out.println("Method dengan 1 parameter = "+i);
7      }
8      public void Tampil(int i, int j){
9          System.out.println("Method dengan 2 parameter = "+i+" & "+j);
10     }
11     public void Tampil(String str){
12         System.out.println(str);
13     }
14
15     public static void main(String a[]){
16         Overloading objek = new Overloading();
17         objek.Tampil();
18         objek.Tampil(8);
19         objek.Tampil(6,7);
20         objek.Tampil("Hello world");
21     }
22 }
```

- **Overriding** method adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya. Contoh overriding method dapat dilihat pada subclass “Mobil” yang mendefinisikan kembali method keterangan() dan hapus() dari class “Kendaraan”.